Task 1:

Exercise 6.1.2:

Movies (title, year, length, genre, studioName, producerC#)
StarsIn (movieTit1e, movieYear, starName)
MovieStar (name, address, gender, birthdate)
MovieExec (name, address, cert#, netWorth)
Studio (name, address, presC#)

a) Find the address of MGM studios:

```
SELECT address
FROM Studio
WHERE name = 'MGM';
```

b) Find Sandra Bullock's birthdate:

```
SELECT birthdate
FROM MovieStar
WHERE name = 'Sandra Bullock';
```

c) Find all the stars that appeared either in a movie made in 1980 or a movie with "Love" in the title:

```
SELECT starName
FROM StarsIn
WHERE movieYear = 1980
 OR movieTitle LIKE '%Love%';
```

d) Find all executives worth at least $10,000,000:

```
SELECT name
FROM MovieExec
WHERE netWorth >= 10000000;
```

e) Find all the stars who either are male or live in Malibu (have string Malibu as a part of their address):

```
SELECT name
FROM MovieStar
WHERE gender = 'M'
OR address LIKE '%Malibu%';
```

Exercise 6.1.3:

Product (maker, model, type)
PC (model, speed, ram, hd, price)
Laptop (model, speed, ram, hd, screen, price)
Printer (model, color, type, price)

a) Find the model number, speed, and hard-disk size for all PC’s whose price is under $1000:

```
SELECT model, speed, hd
FROM PC
WHERE price < 1000;
```

b) Do the same as (a), but rename the speed column gigahertz and the hd column gigabytes:

```
SELECT model, speed AS gigahertz, hd AS gigabytes
FROM PC
WHERE price < 1000;
```

c) Find the manufacturers of printers.

```
SELECT DISTINCT maker
FROM Product
WHERE type = 'Printer';
```

d) Find the model number, memory size, and screen size for laptops costing more than $1500:

```
SELECT model, ram, screen
FROM Laptop
WHERE price > 1500;
```

e) Find all the tuples in the Printer relation for color printers. Remember that color is a boolean-valued attribute:

```
SELECT *
FROM Printer
WHERE color = TRUE;
```

f) Find the model number and hard-disk size for those PC ’s that have a speed of 3.2 and a price less than $2000:

```
SELECT model, hd
FROM PC
WHERE speed = 3.2
AND price < 2000;
```

Exercise 6.1.4:

Classes (class, type, country, numGuns, bore, displacement)
Ships (name, class, launched)
Battles (name, date)
Outcomes (ship, battle, result)

a) Find the class name and country for all classes with at least 10 guns:

```
SELECT class, country
FROM Classes
WHERE numGuns >= 10;
```

b) Find the names of all ships launched prior to 1918, but call the resulting column shipName:

```
SELECT name AS shipName
FROM Ships
WHERE launched < 1918;
```

c) Find the names of ships sunk in battle and the name of the battle in which they were sunk.

```
SELECT ship, battle:
FROM Outcomes
WHERE result = 'sunk';
```

d) Find all ships that have the same name as their class:

```
SELECT name
FROM Ships
WHERE name = class;
```

e) Find the names of all ships that begin with the letter “R.”:

```
SELECT name
FROM Ships
WHERE name LIKE 'R%';
```

f) Find the names of all ships whose name consists of three or more words (e.g., King George V):

```
SELECT name
FROM Ships
WHERE LENGTH(name) - LENGTH(REPLACE(name, ' ', '')) >= 2;
```

Task 2:

Had issues with Atlassian, since I signed up under my school email I am unable to commit to the repository. I can try Github.

Task 3:

Game database, me and Nathan are working on a puzzle game for our game design course this semester. It will include a leaderboard database.

- This project will be useful for other games as the actual interfacing and adding data from the game to the database will be the same between all games. The only difference will be what type of data. Puzzle and arcade games will store high scores, fighting games might store ranking information (ELO, MMR, etc)
- Storing high scores and when this score was achieved, this can be used to track top performers.
- Client and server communication needs to be secure and stable. Clients will never communicate with the database directly, only the server will. There will need to be a layer/api for the server to communicate with the database.

Business:

Businesses can use a database to help keep track of employees and other information within the company.

- Other businesses could use this database for their company, as information about employees and expenses are not unique to just one business.
- Managers could check when employees clock in, and can even average out when they clock in or out. Expenses could be easily tracked using other software as long as it's stored in the database.
- This would require some form of API to communicate so something like a REST api could allow for the data from the database to be used for websites and other web services.